Our Docket No.: 42P12483

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:

Jamil

Application No.: 10/039,060

Filed: January 2, 2002

For: Transfer of Cache Lines On-Chip Between Processing Cores in a Multi-Core System

Examiner: Lane, John A.

Art Group: 2188

Commissioner of Patents
P.O. Box 1450
Alexandria, VA 22313-1450

DECLARATION UNDER 37 CFR 1.131 IN SUPPORT OF PRIOR INVENTION

Sir :

I, Cameron B. McNairy declare:

1. I am an inventor of the claims of the above-captioned patent application ("the Application") and an inventor of the subject matter described therein.

2. Prior to July 26, 2001, the filing date of U.S. Patent Application Publication No. 2003/0023794 cited in an Office Action mailed November 26, 2004, the invention claimed in the Application had been conceived and reduced to practice in the United States.

3. Attached Exhibit A is a redacted copy of an invention disclosure form describing the design of the System and Method To Transfer Cache Lines On-Chip Between Processing Cores in a Multi-Core Processor, and establishes that the subject matter claimed in the Application had been reduced to practice in the United States prior to July 26, 2001.

I further declare that all statements made herein of our own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements are made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both under section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application of any patent issuing thereon.

Dated: 7 FEB, 2005

Cameron B. McNairy

Rev. 15, 8/00

EXHIBIT A

It is proving increasingly difficult to extract more fine grain, instruction level parallelism for enhancing the performance of traditional single thread microprocessors, especially for enterprise and internet applications, which are more memory, than computation bound. To solve this problem, Chip multiprocessors (CMP) are viewed as an evolutionary step forward to extend performance and cost/performance of processors through providing coarse grain, thread level parallelism capability on a chip. Furthermore, the complexity involved in designing an ever wider superscalar machine, suggests a value proposition of a CMP built from already designed processor cores from a time to market perspective.

The inventions disclosed here describe methods of transferring modified cache lines between cores in a CMP. These methods are employed when one core requests data that is only present in a modified state within another core's private cache. Thus, to ensure correct cache coherence, the data needs to be provided from that other core's cache. These methods improve the memory subsystem performance of a CMP, by allowing these line transfers to occur on chip without requiring access to the external interconnect such as a Front Side System Bus. These methods also are applicable for use with processor core and cache/bus microarchitectures that were designed for traditional single core processors. As such these inventions provide Intel with a low cost means of implementing a performance-optimized cache coherent CMP without significant investment in a custom CMP oriented core, or a special CMP oriented multi-core interconnect infrastructure.

In the figures below, illustrating the inventions, Core 0 want to access a location, that has already been modified by Core 1 and exists in a modified state in Core 1's private cache. This requires the modified line to be transferred from Core 1 to Core 0. In the first option, illustrated in Figure 1, the modified line is written back into a shared cache from the core owning the line, and subsequently read back out of the shared cache again to satisfy the requesting core. This option reuses core line write-back and core line-fill functionality that anyway exists to evict line on replacements and fill lines on misses. In the second option, illustrated in Figure 2, the requested line is written back into a fill buffer, consisting of fill slots for each outstanding request. The line is then both supplied to the core, and written back into shared cache, similar to a completing external request transaction. The third option, illustrated in Figure 3, is to multiplex the write back datapath from each core into the fill datapath to each core, and essentially uses a multiplexor to bypass the data written out of one core into the requesting core. The other input into the multiplexor is the standard line fill path into the core from the shared cache.

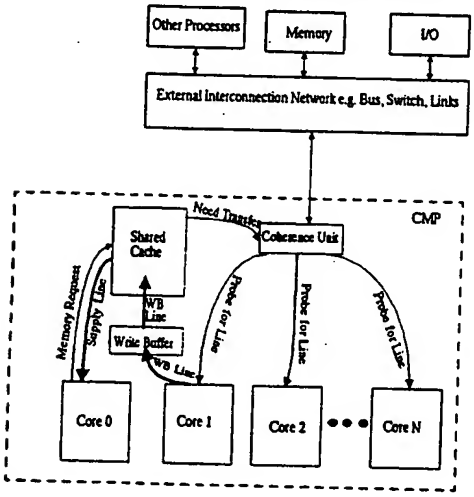

Figure 1

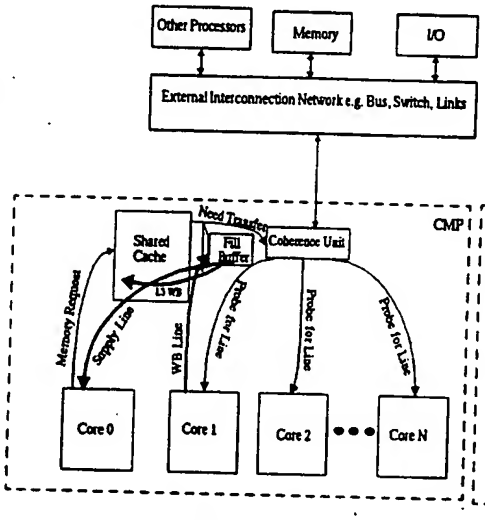

Figure 2

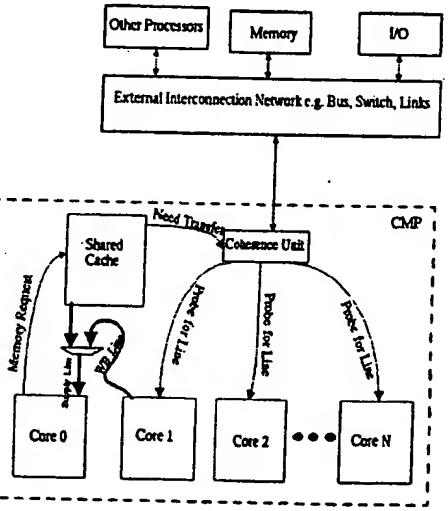

Figure 3

New functionality, data structures, and protocols are needed for all three options, some common to all options, some unique to each option. In all three options, the original request is rejected once it is determined that the requested data is present in another core's cache. The request is periodically retried to check if the line transfer has completed. All three options would also require the coherence unit logic to track the misses that require internal snoop probes of the other core, as opposed to probes initiates from external SB transactions. The coherence unit will arbitrate between these internal probes and probes arriving from the external interconnect for system coherence maintenance. The coherence unit implements a fairness and forward progress scheme to ensure that internal and external probes all progress fairly and do not starve each other. A signaling interface would be required between the shared cache control logic and the coherence unit to allow the shared cache to request an internal transfer probe and for the coherence unit to indicate to the shared cache to hold off further requests, when its probe queues are full.

All the options would also need to allow for a replacement of the line being requested in the other core's private cache while the transfer request was still in progress. If the probe finds that the requested line is being victimized, the request is de-allocated from the coherence unit, and no requests are made to the probed core to write back the line, since a victim write back is already under way. Since the request will periodically reissue into the shared cache anyway, no special case is needed for the request to see the results of the victim write back into the shared cache. If an external probe to the requested line precedes the internal probe, when the internal probe finally issues, it will no longer find the line in the other core. However, the external snoop would also invalidate the shared cache line. So, subsequently when the request is reissued into shared cache, it will miss, and proceed to the external network for fulfillment.

Of course, each of these options also had its own particular requirements. The first option for example requires a write buffer to temporarily store the written back line. The second option required that the fill buffer write port be multiplexed between the data input from the external network and write back data path from each core. Each fill buffer entry write enable would also need to be independent to allow for a simultaneous data return from the external network and a write back from a core. The third option would require each core's line write back datapath to be routed into a multiplexor that connects to the fill datapath into every other core.

The closest prior art we are aware of is the IBM Power 4, which is a 2-core CMP with a shared L2 cache, with each core containing private L1 caches. To our knowledge, IBM's public disclosures have not disclosed whether or how the cores transfer cache lines between the private core

caches. Any processor manufacturer implementing CMP products may use this invention, and infringement may be detected by monitoring the external interconnect to observer whether or not all inter-core traffic appears externally.